IBM Technical Newsletter

File Number	1410/7010-36
Re: Form No.	C28-0319
This Newsletter No.	N28-1077
Date	August 5, 1963
Previous Newsletter Nos.	None

IBM 1410/7010 SYSTEM MONITOR

This newsletter contains replacement pages reflecting an expansion of the specifications described in the publication, IBM 1410/7010 Operating System; System Monitor: Preliminary Specifications, Form C28-0319.

Changes in text are indicated by a vertical bar to the left of the text. Changes in figures are indicated by a (●) to the left of the figure number. A vertical bar to the left of headings on pages 7, 13 and 30 indicates a deletion.

International Business Machines Corp., Programming Systems Publications, P. O. Box 390, Poughkeepsie, N. Y.

PRINTED IN U.S.A.

Introduction

Purpose of this Publication

This publication describes the use of the 1410/7010 Operating System under control of the System Monitor. Included in this material is information concerning the functions and components of the System Monitor, its relationship to the Operating System, and considerations for writing programs to run under control of the System Monitor.

Purpose of the System Monitor

The System Monitor performs the control functions for the 1410/7010 Operating System. Some of the major functions performed for programs within the Operating System are as follows:

Assignment of input/output units.

Program loading, including relocation of programs, and linkage between programs and subroutines that were independently written and compiled.

Programmed transition from run-to-run and job-to-job.

Advantages of the System Monitor

Efficient Machine Operation: The primary goal of the Operating System is the efficient use of machine time. The System Monitor works toward this goal by facilitating quick and automatic transition from run-to-run and job-to-job, and by minimizing and simplifying operator intervention.

Definition of Programming and Operating Standards: An additional advantage provided by use of the System Monitor is the establishment of standards for programming and for machine-room operations. These standards result in better communication between the individual programmers within an installation and between programmers and machine-room personnel. Consequently, time-savings can be realized in the planning and integration of programs, and also in the operation of those programs. Furthermore, this establishment of standards facilitates the exchange of programming with any other installation that uses the 1410/7010 Operating System.

Segmentation of Programs: Because of the System Monitor's ability to relocate and link programs in storage, programs can be written and tested in logical segments. When it is time to integrate the various segments into a complete program, the System Monitor will efficiently assign storage locations for each of the segments and link the segments together. This facility also enables an installation to create a collection of subroutines that can be incorporated into any number of main programs.

Capabilities for TELE-PROCESSING® Systems: The System Monitor can provide, at the option of each installation, control facilities for IBM TELE-PROCESSING® Systems. Detailed information concerning this type of application is contained in the publication, *The TELE-PROCESSING Supervisor,* Form C28-0321.

Prerequisite and Related Literature

The publication, *1410/7010 Operating System: Basic Concepts,* Form C28-0318, is prerequisite literature for this publication. The reader is assumed to be familiar with both the terminology and concepts defined by that publication.

Related literature includes the following publications, which are recommended to the reader who wishes detailed information concerning all the elements and functions of the 1410/7010 Operating System:

System Generation, Form C28-0320

The Basic Input/Output Control System, Form C28-0322

The TELE-PROCESSING Supervisor, Form C28-0321

Utility Programs, Form C28-0325

The Generalized Tape Sorting Program, Form C28-0324

Random-Processing Scheduler, Form C28-0323

Autocoder, Form C28-0326

FORTRAN, Form C28-0328

COBOL, Form C28-0327

Form C28-0319
Page Revised August 5, 1963
by TNL N28-1077

Basic Principles of the System Monitor

Logical Structure of the System Monitor

The System Monitor consists of three major elements. The first of these elements comprises the control routines that remain in core storage at all times. This element is called the *Resident Monitor.* The second element consists of routines that perform functions related to the transition from run-to-run and job-to-job. This element, which is called the *Transitional Monitor,* is brought into storage when such transitional functions are required. The analysis of control cards for the System Monitor is one of the major functions performed by the Transitional Monitor. The third element, which is called the *Linkage Loader,* performs the functions required to prepare relocatable programs for execution.

Functions of the Resident Monitor

The following information outlines the major functions performed by routines within the Resident Monitor. More detailed information concerning the use of these functions can be found in later sections of this publication.

INPUT/OUTPUT CONTROL SYSTEM

The Input/Output Control System (IOCS) is created during System Generation to meet the requirements of each installation using the Operating System. This IOCS, which is termed the *Resident IOCS,* is an integral part of the Resident Monitor. It contains the basic routines, such as those required for error checking and channel scheduling, that perform functions common to all input/output operations. (For detailed information concerning the Resident IOCS, see the publication, *Basic Input/Output Control System,* Form C28-0322.)

UNIT-RECORD ROUTINES

The Resident Monitor contains three routines to handle input/output operations for unit-record equipment. One of these routines is for card input, another for card output, and the third is for printer output. At the installation's option, each of these unit-record functions can be performed with magnetic tape. For card output and printer output functions, the installation specifies at System Generation time whether that function is to be performed with magnetic tape, unit-record equipment or not at all.

ASSIGNMENT ROUTINE

Facilities for performing functions related to the assignment of input/output units for program use are included in the Resident Monitor and in the Transitional Monitor. The Resident Monitor contains tables of information concerning the installation's input/output equipment, and facilities for providing this information as it is required during the execution of programs. The Transitional Monitor performs the analysis of control-card information related to input/output assignment, and coordinates this information with the functions of the Resident Monitor's Assignment Routine.

END-OF-PROGRAM ROUTINE

The End-of-Program (EOP) Routine analyzes each end-of-program situation to determine the function to be performed next. If it is a *Normal EOP* (successful completion of the program being executed), the Transitional Monitor is brought into storage to analyze the control cards that define the next run.

If the end-of-program situation is caused by program failure (this situation is called *Unusual EOP*), the EOP Routine can preserve the status of core storage by writing onto tape an image of the contents of storage. The Transitional Monitor is then brought into core storage to analyze control cards from the Standard Input Unit.

LOAD ROUTINE

The Load Routine brings into storage those programs that are ready for immediate execution. Such programs, which are said to be in *absolute format,* include programs that have been prepared for execution by the Linkage Loader and programs that are on the System Operating File. (The Linkage Loader, for example, is in absolute format and resides on the System Operating File. It is one of the programs that is loaded directly into core storage by the Load Routine.)

The Load Routine also performs the searching functions that are required during the execution of a job. For example, this routine locates the next phase of a multiphase program and brings that phase into storage. Note that the *next* program phase need not be the next in numerical sequence. Each program phase, at its completion, specifies to the Load Routine which phase is to be located. For example, Phase 1 of a program could issue a request for either Phase 2 or any other phase, basing its choice on the result of the processing in Phase 1.

COMMUNICATION REGION

The Communication Region is a collection of data areas that contain various types of control information. Some of these areas contain information used for communication between various IBM programs within the Operating System; others contain information of value to the user's programs operating under control of the System Monitor.

For example, one area of the Communication Region is used to store the current date. This area can be addressed by any program. The Resident IOCS uses this area to determine the date for writing and checking tape labels. The user's programs can address this area for such purposes as creating a header record for printer output.

All modifications to the information in the Communication Region are performed through the Resident Monitor.

WAIT-LOOP ROUTINE

The Wait-Loop Routine provides a means for programs to suspend processing until the machine operator performs a specified function, such as changing the type of forms being used for the printer. A program can write a message on the console typewriter, branch to the Wait-Loop Routine, and resume processing after the operator signals the Resident Monitor that the function specified by the message has been completed.

CONSOLE INQUIRY ROUTINE

Facilities for responding to Console Inquiries are included in both the Resident Monitor and the Transitional Monitor. The Resident Monitor accepts Console Inquiries that can be serviced during a run, such as the signal from the operator to exit from the Wait-Loop Routine. The Transitional Monitor accepts Console Inquiries related to functions performed between runs, such as a signal to begin reading from the Alternate Input Unit.

The Console Inquiry Routines (in both the Resident and Transitional Monitors) serve as a means of communicating control information to the System Monitor.

Functions of the Transitional Monitor

The following information outlines functions performed by routines within the Transitional Monitor. The preceding information concerning functions of the Resident Monitor also includes information related to functions for which both the Resident and Transitional Monitors provide facilities.

CONTROL-CARD INTERPRETATION ROUTINE

When the Transitional Monitor is brought into storage, one of its major functions is the interpretation of control cards directed to the System Monitor to define the next run (or job). (These cards include an identification field containing the characters MON$$ and are termed Monitor control cards.) For this function, the Transitional Monitor contains a Control-Card Interpretation Routine. This routine analyzes the Monitor control cards and, in accordance with their specifications, gives control to whichever elements of the System Monitor are required to prepare and execute the next run.

JOB ROUTINE

The Job Routine performs functions related to the initialization of the Resident Monitor for the next job. This includes resetting various program switches, clearing certain areas in the Communication Region, and insuring that all IOCS functions for the previous job are completed.

Functions of the Linkage Loader

The Linkage Loader converts relocatable program elements into absolute format and in the process resolves all symbolic linkages and references into machine-language instructions and addresses.

For each program requiring these functions, the Linkage Loader produces a file, in absolute format, consisting of all the program elements that are to be executed as one logical program unit. (The final program unit is considered to be *executable,* since it is ready to be loaded directly into core storage and given processing control. Also note that a single executable program can be divided into phases which are loaded and executed one after another.) The file produced by the Linkage Loader is called the *Job File.* This file is brought into core storage by the Load Routine contained in the Resident Monitor.

The basic program unit with which the Linkage Loader performs its functions is termed a *subprogram.*

A subprogram normally consists of the relocatable output from one compilation by a language processor. The Linkage Loader can construct a program in absolute format from one or more subprograms. Selection of the subprograms that are to be converted into a single executable program is determined by control cards from the Standard Input Unit, or by control information imbedded in the subprograms being processed.

Execution of the Linkage Loader is requested by a Monitor control card from the Standard Input Unit. This permits the use of the Linkage Loader as an independent program, in the same manner as a compiler or sorting program, and offers the user considerable flexibility in the use of the Linkage Loader to meet the requirements of various jobs. (For example, the Linkage Loader can be directed to create a Job File that is, in effect, a library of programs in absolute format. These programs can then be executed in the same job, and the file can also be saved for execution of the programs in later jobs.)

The Linkage Loader also has the ability to include the Snapshot Program into the program that is being converted to absolute format. This optional feature is requested by a control card from the Standard Input Unit, or by a control statement imbedded in a subprogram.

An additional feature of the Linkage Loader is the ability to incorporate patches into a previously compiled subprogram.

Input/Output Files for the System Monitor

To perform the control functions for programs within the Operating System, the System Monitor requires a group of input/output files. These files, which are described below, can be assigned to various configurations of input/output units.

System Operating File

The System Operating File contains the programs within the Operating System that are in absolute format. This necessarily includes the System Monitor, and can include the language processors (with associated data files), the Utility Programs, and the Sort Definition Program. (The first element of the System Operating File is a routine, called the *Bootstrap,* that initially loads the Resident Monitor into core storage.) The System Operating File is created at the time of System Generation, and is constructed according to the program requirements of each installation. When this file is created, the user may add other programs, such as sorting programs produced by the Sort Definition Program. The System Operating File can also include programs in relocatable format. Such programs are contained within a contiguous section of the System Operating File and constitute the System Library (which is discussed below). The System Operating File may be on either tape or disk.

System Library File

The System Library File contains relocatable programs. This file is one of the input sources for the Linkage Loader. The Library may be on either tape or disk. (If the Library is on tape, it can be on the same tape as the System Operating File or it can be on a separate reel.) An installation can have any number of Libraries, but only one is designated as the System Library File for a particular run of the Linkage Loader.

Standard Input Unit

The Standard Input Unit contains the file of control information for the System Monitor. Source programs for the language processors, relocatable programs to be processed by the Linkage Loader, and input data for the user's programs may also be placed in the Standard Input Unit. At the option of each installation, an Alternate Input Unit may also be designated. Control information can be submitted to the System Monitor through either unit. This configuration permits a high-priority job in the Alternate Input Unit to break into the previously established sequence of jobs in the Standard Input Unit. The Standard Input Unit can be a card reader or a magnetic tape unit. The Alternate Input Unit can also be either a card reader or a magnetic tape unit.

Standard Print Unit

The Standard Print Unit is used for all printer output from IBM programs within the Operating System. It is also available for use by the installation's programs. This unit can be either the IBM 1403 Printer or a magnetic tape unit. If it is a tape unit, it can be the same unit as the Standard Punch Unit, which is described below.

Standard Punch Unit

The Standard Punch Unit is used for all punched-card output from IBM programs within the Operating System. It is also available for use by the installation's programs. This unit may be either the IBM 1402 Card Read Punch or a magnetic tape unit. If it is a tape unit, it can be the same unit as the Standard Print Unit.

Autocoder includes language statements that directly relate to the use of Communication Symbols.

LINKAGE SYMBOLS

Linkage Symbols are established by two types of cards: the Title card of a subprogram and Definition cards that are included in a subprogram. (Autocoder includes the DEFIN statement for the creation of Definition cards.)

Each time the Linkage Loader encounters one of these cards, it places into a table the characters that have been declared as a Linkage Symbol and the address assigned by the compiler to the Linkage Symbol. (The address is placed into the table after it has been relocated.) For example, a Title card declares the name of the subprogram as a Linkage Symbol, and the origin point of the subprogram (after relocation) becomes the address assigned to that Linkage Symbol. This procedure enables a subsequent subprogram to refer to a previously-processed subprogram by means of the Linkage Symbol established for the name of the previous subprogram.

A Linkage Symbol can appear in a subprogram in either of two formats:

1. A symbol consisting of one to ten alphameric characters, such as the name of a subprogram. (The first character must be alphabetic.)

2. A five-character symbol consisting of four alphameric characters followed by a slash (ABCD/). The first character must be alphabetic.

References to Linkage Symbols of the format described in item 1 above, are made by *DCWS* and *DCWF* cards. (COBOL and FORTRAN automatically generate these cards in accordance with the requirements of their source programs. Autocoder includes language statements for the generation of these cards.) When the Linkage Loader encounters a DCWS card, it supplies the subprogram with a branch instruction containing the address assigned to the Linkage Symbol specified by that DCWS card. The DCWF card causes the Linkage Loader to supply the subprogram with a five-character constant of the address assigned to the Linkage Symbol.

References to Linkage Symbols of the format ABCD/ can be made either by DCWS and DCWF cards, or by use of the symbol in the A-field or B-field of an instruction. For example, in the Autocoder language, the programmer can use the DEFIN statement to declare WORK/ as a Linkage Symbol that represents the address of a work area. The programmer can then use WORK/ as an operand of an instruction that refers to that work area, such as S + 10, WORK/. Such an instruction can be included in any other subprogram that is to be processed by the Linkage Loader during the same run as the subprogram that declared the Linkage Symbol.

For each Linkage Symbol of the format ABCD/, the Linkage Loader supplies the subprogram with the address equated to that symbol in the Linkage Loader's symbol table.

SYSTEM SYMBOLS

All System Symbols have the format, /ABC/. These symbols can be used in the same manner as Linkage Symbols of the format, ABCD/. (The following section of this publication, "Programming Considerations for use of the System Monitor," contains information concerning System Symbols that can be used for direct reference to elements within the Resident Monitor, such as the Communication Region.)

LISTING OF COMMUNICATION SYMBOLS

The Linkage Loader produces a listing of all Communication Symbols used during the processing of subprograms. Each of the symbols is listed with the core-storage address that it represents. Thus, the listing provides a means for determining the location of program elements after they have been relocated.

Common Data Areas

In addition to the use of Linkage Symbols, separately compiled subprograms can communicate with each other by the establishment of a common data area. As mentioned earlier, the Linkage Loader applies a special *downward* relocation to addresses that refer to such a data area. The term *downward relocation* derives from the fact that common data areas are compiled from a particular location downward to a lower location, while other parts of a subprogram (such as its instructions) are compiled from a particular location up to a higher location.

Multiphase Programs

The Linkage Loader has the ability to accept control information specifying the division of a single executable program into various phases. (Each phase must consist of at least one subprogram.) This information is submitted through the Standard Input Unit

with the other control cards for the Linkage Loader. The control cards used for construction of multiphase programs are included in the following topic.

Control Cards for the Linkage Loader

The information in this topic includes the functions and formats of the various control cards that can be submitted to the Linkage Loader through the Standard Input Unit.

The general format of control cards for the Linkage Loader is the same as the format used for Autocoder source statements:

Columns 16-20 (the *Operation* field) contain the name of the card, identifying its function.

Columns 21-72 (the *Operand* field) are used to specify information related to the function indicated in columns 16-20.

Note that control cards for the Linkage Loader are *not* Monitor control cards; that is, they do not contain the identification MON$$. Rather than directing the general flow of processing during a batch, Linkage Loader control cards direct the processing to be performed during one run of the Linkage Loader (in much the same manner as control cards direct the processing for one run of a sorting program).

The following list contains the names of the various control cards for the Linkage Loader; the cards are described below in the order of this list. The TITLE and DEFIN cards are described in the publication, "IBM 1410/7010 Operating System, Autocoder: Preliminary Specifications," Form C28-0326. Those cards preceded by an asterisk (*) can be contained within a subprogram (in addition to being submitted through the Standard Input Unit).

* CALL
CALLN
CALLP
PHASE
* BASE1
* PRTCT
* BASE2
SNAP
INPUT
* TITLE
* DEFIN

The CALL Card

The operand of the CALL card is the name of the subprogram to be processed by the Linkage Loader. This card is used for a subprogram located on either the System Library File or the Go File. A subprogram located in the Standard Input Unit does not require a CALL card; its Title card serves the function of the CALL card. When the Linkage Loader encounters a CALL card, it adds the specified name to a list. At the time the Linkage Loader is ready to begin processing the subprograms on this list, it searches the Go File and then the System Library File to find them.

It is important to note that this searching is performed by reading the names of the programs on the Go File and the System Library File and comparing them to the names on the list of requested subprograms. This technique minimizes the time required for searching.

Because the Linkage Loader employs this technique in order to minimize search time, the final sequence of subprograms in core storage is not necessarily the same sequence as the CALL cards that specified the requests for those subprograms. (The CALLN card, which is described later, can be used to control the relative positioning of subprograms in core storage.)

For example, assume that the following subprograms are arranged sequentially on the System Library File: SP1, SP2, SP3, SP4, and SP5. Assume also that CALL cards were read from the Standard Input Unit in the following order: CALL SP3, CALL SP1, CALL SP4. The Linkage Loader begins its search of the System Library File and locates the Title card for SP1. It checks this name against the names on the list and finds that there is a request for SP1; therefore, it processes SP1, and places it on the Job File. The Linkage Loader then searches forward in the System Library File and locates the Title card for SP2. It checks the list, finds no request for SP2, and continues in the search. The Title card for SP3 is located in the Library and checked against the list. The Linkage Loader finds that there is a request for SP3 and, therefore, processes it and places it on the Job File behind SP1. This procedure continues until all the subprograms on the list have been located, processed, and placed on the Job File.

In the example above, the final sequence of the subprograms would be SP1, SP3, SP4 – the order in which they were located on the Library. (The relative location of subprograms in storage is referred to as the *memory map*. This term will be used to present examples that illustrate control of the positioning of a subprogram.) The relocation factor for the three subprograms was adjusted as follows: SP1 was given an origin point of Basc Zero and all upward relocation for SP1 was performed with a relocation factor equal to Base Zero; SP3 was given an origin point of Base Zero plus the size of SP1 and all upward relocation for SP3 was performed with a relocation factor equal to Base Zero plus the size of SP1; SP4 was given an origin point of Base Zero plus the combined size of

SP1 and SP3 and all upward relocation for SP4 was performed with a relocation factor equal to Base Zero plus the combined size of SP1 and SP3.

IMBEDDED CALLS

In addition to requests for subprograms specified by CALL cards, the Linkage Loader interprets DCWS and DCWF cards as requests for subprograms. (Since these cards are contained within subprograms, this type of request is termed an *imbedded call.*)

The imbedded call implied by a DCWS or DCWF card is related to the establishment (by a Title card) of the name of a subprogram as a Linkage Symbol. When the Linkage Loader encounters an imbedded call, it checks its symbol table to determine whether the name of the called subprogram has been established yet as a Linkage Symbol. If it has (meaning that the subprogram has already been processed, since its Title card established the symbol), the Linkage Loader replaces the imbedded call with either a branch to the address assigned to the Linkage Symbol, or with a five-character constant containing that address (depending on whether the imbedded call is a DCWS or DCWF).

If the called subprogram has not yet been processed, the Linkage Loader places the name of the subprogram on its list of requests (the same list used for requests specified by CALL cards). When the Linkage Loader later locates and processes the subprograms on this list, it generates cards that supply the previous imbedded calls with the branch addresses and constants that are determined by the processing of the Title cards of the subprograms.

NOTE 1: The Linkage Symbol established by the Title card of a subprogram represents the *origin* point of that subprogram. Autocoder, in addition, provides the DEFIN statement for establishing Linkage Symbols to represent *any* point in a subprogram. These Linkage Symbols can also appear in DCWF and DCWS cards, and are treated by the Linkage Loader in the same manner as DCWF and DCWS cards that contain the name of a subprogram. It is important to note, however, that a subprogram is located and processed only if it has been called by the name contained in its Title card. Therefore, if subprogram JOEL contains an entry point represented by the Linkage Symbol JANE/, the card DCWS JANE/ cannot be completely processed unless a call is made for JOEL, either by a CALL card or an imbedded call.

NOTE 2: Linkage Symbols established within a particular phase of a multi-phase program can be referred to by subprograms within the same phase and succeeding phases, but they cannot be referred to by preceding phases. For example, Phase 1 cannot contain a DCWS card containing the Linkage Symbol IRMA/ if that symbol is not established (by a Title card or Definition card) until Phase 2.

The following rules apply to the calling of programs:

1. Rules of precedence for memory maps:
   a. The first subprograms in core storage will be those read from the Standard Input Unit (in the order read).
   b. The second group of subprograms in core storage will normally be those read from the Go File (in the order read).
   c. The third group of subprograms in core storage will normally be those read from the System Library File (in the order read).
   d. If the subprograms in steps b and/or c contain imbedded calls, then steps b and c will be repeated, in that order, until all calls are satisfied.

2. Imbedded cells are considered to be for subprograms in the System Library File.
   If the subprogram requested by an imbedded call is on the Go File, it must be requested by a CALL card.
   NOTE: The CALL card may be omitted if the requested subprogram follows the requesting subprogram on the Go File.

3. Subprograms placed in the Standard Input Unit do not require a CALL card; their Title card serves this function.

EXAMPLES

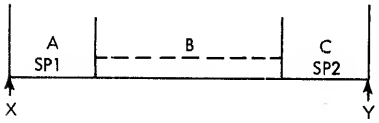
To illustrate the use of the various control cards for the Linkage Loader, examples are presented after the descriptions of the functions of those cards. For the convenience of the reader, a standard format is employed for these examples.

Each example begins with a description of the type of operation to be performed by the Linkage Loader (such as construction of a multiphase program), and includes a description of the situation in which this operation is to be performed (location of the subprograms, whether or not they contain imbedded calls, etc.).

Following the definition of the situation is a sequential list of the control cards that are to be placed in the Standard Input Unit in order to meet the requirements of the situation. (The Monitor control card that initiates execution of the Linkage Loader is assumed to immediately precede the control cards for the Linkage Loader.) The Linkage Loader determines that it has come to the end of its control cards when it finds that the next card in the Standard Input Unit is not in a format that the Linkage Loader recognizes.

(The next card must be a Monitor control card. The Appendix of this publication contains a sample control-card deck, illustrating the relative positions of Linkage Loader control cards and Monitor control cards.) Following the list of control cards in each example is a description of the Linkage Loader's procedures as it reads the cards.

The last item in the examples is a diagram of the memory map that results from the Linkage Loader's processing. The following format is used for these diagrams:



The area from point "X" to point "Y" represents the area of core storage used for the subprograms that were just processed. When the name of only one subprogram appears in an area (such as SP1 in area "A"), the diagram indicates that this subprogram will be definitely positioned in this area, relative to any other subprograms in core storage. (The diagram above indicates that SP1 will definitely be the lowest subprogram in core storage, and SP2 will be the highest subprogram.) The dotted line through area "B" indicates that this area of storage will not be loaded for this particular phase. (This type of area is used in examples of multiphase programs.)

The following terminology is employed in the examples:

1. SIU: The Standard Input Unit.
2. Library: The System Library File.
3. Request list: The list on which the Linkage Loader places the name of a subprogram when it is called.
4. Flag: The indicator that the Linkage Loader sets for a name on the request list after the subprogram of that name has been processed.
5. Title Card for SPX: A term used in the list of control cards at the SIU. The term indicates that subprogram SPX is placed in the SIU at this point, relative to any other cards for the Linkage Loader.

EXAMPLE 1

This example illustrates the procedure for directing the Linkage Loader to process a subprogram from the Library. The subprogram (SP1) constitutes a self-contained, single-phase program that contains no imbedded calls for other subprograms.

Only one control card is necessary: CALL SP1. The Linkage Loader will locate SP1 on the Library, process it, and place it on the Job File.

EXAMPLE 2

This example illustrates the procedure for directing the Linkage Loader to construct a single-phase program from three subprograms: SP1, SP2, and SP3. (One of the three must be a primary subprogram, the other two can be secondary.) In this example, SP1 is on the Library, SP2 is in the SIU, and SP3 is on the Go File. None of the subprograms contain imbedded calls.

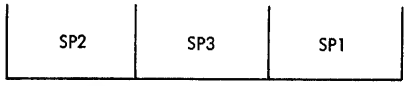
The sequence of cards in the SIU is as follows:

```
CALL    SP1
Title Card for SP2
CALL    SP3
```

The Linkage Loader proceeds as follows:

1. CALL SP1: The name SP1 is placed on the request list.
2. Title card for SP2 (followed by other cards of SP2): SP2 is processed and the name SP2 is added to the request list, with a flag indicating that SP2 has been processed.
3. CALL SP3: The name SP3 is added to the request list. The Linkage Loader resumes reading from the SIU and finds no further control cards. It begins searching the Go File, locates SP3, and processes it. Next, the Library is searched. SP1 is located and processed.

The memory map is as follows:



EXAMPLE 3

This example illustrates the construction of a single-phase program from three subprograms, one of which is processed because of an imbedded call. SP1 is on the Library and contains an imbedded call for SP3; SP2 is on the Go File.

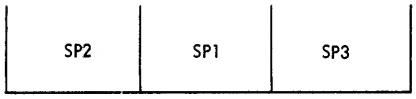
The sequence of cards in the SIU is as follows:

```
CALL    SP1
CALL    SP2
```

The Linkage Loader proceeds as follows:

1. CALL SP1: The name SP1 is placed on the request list.
2. CALL SP2: The name SP2 is placed on the request list. The Linkage Loader resumes reading from the SIU and finds no further control cards. It begins searching the Go File, locates SP2, and processes it. Next, the Library is searched. SP1 is located and processed. During the processing of SP1, the Linkage Loader finds the imbedded call for SP3. At that time, the name SP3 is placed on the request list. When the processing of SP1 is completed, the Linkage Loader checks the request list, finds the unsatisfied request, and resumes the search of the Library. SP3 is located and processed.

The memory map is as follows:



The CALLN Card

The function of the CALLN card is essentially the same as the CALL card, except that the CALLN card directs the Linkage Loader to immediately locate and process the specified subprogram (CALLN means Call Now). This card can be used to control the memory map.

The relative position of the subprogram specified by a CALLN card is established in accordance with the following Linkage Loader procedure:

1. All subprograms specified by previous CALL cards are located and processed.

2. Any imbedded calls resulting from the processing of those subprograms are also processed (except imbedded calls for the subprogram named in the CALLN card).

3. The subprogram specified by the CALLN card is processed after the subprograms specified in 1 and 2, above.

4. Any imbedded calls resulting from the processing of the subprogram specified by the CALLN card are added to the list of requested subprograms (if the imbedded calls have not been satisfied by the processing already performed).

The following rules apply to the use of CALLN card:

1. Rules of precedence for memory maps:
   a. All subprograms called prior to the subprogram named by the CALLN card will be the first subprograms in core storage. (The relative order of those subprograms is determined in accordance with the rules of precedence for the CALL card.)
   b. The subprogram named by the CALLN card will be placed in core storage following the subprograms specified in 1a, above.

2. If a subprogram located in the Standard Input Unit is to be processed in accordance with the CALLN procedure, it must be immediately preceded by a CALLN card containing the name of that subprogram.

EXAMPLE 1

This example illustrates the construction of a single-phase program from three subprograms, one of which must be positioned between the other two. All three subprograms are on the Library, and none of them contain imbedded calls. The sequence of subprograms on the Library is SP1-SP2-SP3; SP3 must be positioned after SP1 and before SP2.

The sequence of cards in the SIU is as follows:

```
CALL    SP1
CALLN   SP3
CALL    SP2
```

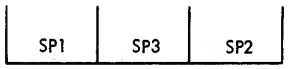
The Linkage Loader proceeds as follows:

1. CALL SP1: The name SP1 is placed on the request list.

2. CALLN SP3: The name SP3 is saved in an area (the CALLN area) used for names of subprograms that appear as the operand of CALLN cards (it is not yet placed on the request list.) The Linkage Loader begins searching for previously-called subprograms. It locates SP1 on the Library and processes it. The Linkage Loader then determines that the request list has been satisfied (SP1 was the only name on the list). The name SP3 is moved to the request list. SP3 is then located and processed.

3. CALL SP2: The name SP2 is placed on the request list. The Linkage Loader resumes reading from the SIU and finds no further control cards. SP2 is then located and processed.

The memory map is as follows:



EXAMPLE 2

This example illustrates the construction of a single-phase program from three subprograms, one of which must be positioned between the other two; another contains an imbedded call for the subprogram that must be placed in the middle position.

The three subprograms are on the Library in the order, SP1-SP2-SP3. SP3 must be positioned after SP1 and before SP2; SP1 contains an imbedded call for SP3.

The sequence of cards in the SIU is as follows:

```
CALL    SP1
CALLN   SP3
CALL    SP2
```

The Linkage Loader proceeds as follows:

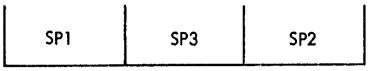
1. CALL SP1: The name SP1 is placed on the request list.

2. CALLN SP3: The name SP3 is saved in the CALLN area. The Linkage Loader begins searching for previously-called subprograms. It locates SP1 on the Library and processes it. During the processing of SP1, the Linkage Loader finds the imbedded call for SP3. It places the name SP3 on the request list. When SP1 is completely processed, the Linkage Loader checks the request list, finds the name SP3, but also determines that SP3 is in the CALLN area. Therefore, the name SP3 is flagged on the request list, and the Linkage Loader makes a note to supply the imbedded call in SP1 with the address of SP3, after SP3 has been

processed. The Linkage Loader then determines that there are no more names on the request list, moves SP3 to the request list, locates SP3, and processes it.

3. CALL SP2: The name SP2 is placed on the request list. The Linkage Loader resumes reading from the SIU and finds no further control cards. SP2 is then located and processed. The Linkage Loader determines that an imbedded call has not yet been supplied with an address (SP1's imbedded call for SP3), checks the symbol table for the address value of SP3 (established by the Title card of SP3), and generates a patch that will place this address value into SP1 at the time the subprograms are loaded into core storage by the Load Routine of the Resident Monitor.

The memory map is as follows:



NOTE: The above memory map is identical to the one in the preceding example. The purpose of this example is to illustrate the procedure used for an imbedded call of a subprogram that is also called by a CALLN card.

The CALLP Card

The CALLP card (Call and Patch) functions in the same manner as the CALLN card, except that the CALLP card directs the Linkage Loader to incorporate patches into the specified subprogram. The patches must immediately follow the CALLP card in the Standard Input Unit.

THE PHASE Card

PHASE cards are used create a multiphase program and to assign a name to that program. Each time the Linkage Loader encounters a PHASE card it performs the following functions:

1. All previous calls (both from control cards and imbedded calls) are processed. The resulting phase is terminated with a record that indicates the entry point of that phase. (This information is used by the Load Routine of the Resident Monitor to initiate execution of the phase after it has been loaded into core storage.)

2. The Linkage Loader creates a header record for the phase, consisting of the name and the number of the phase.

The first PHASE card for a multiphase program specifies the name of that program. Succeeding PHASE cards for the same program have blank operand fields. For the first PHASE card, the Linkage Loader generates a junction with the BASE1 card, examples illustrating the header record containing the name specified in the operand and a phase number of 001. For succeeding PHASE cards of the same multiphase program, the Linkage Loader generates a header record containing the name specified by the first PHASE card, and a phase number determined by incrementing a counter each time a PHASE card with a blank operand is encountered. Therefore, a PHASE card with a blank operand indicates the beginning of the program's next phase, and the header record for this phase contains the next sequential number.

The user may select phase numbers other than the ones assigned by the Linkage Loader. The selected phase numbers must be placed in columns 6, 7, and 8 of the second and successive PHASE cards. The following rules must be observed when selecting phase numbers.

1. The first phase of a program must be designated 001.
2. Subsequent phase numbers must be in ascending order (not necessarily sequential).
3. The phase number 999 must not be used.

NOTE: When the user selects a phase number other than the assigned number, the selected phase number must always be used in calling that phase.

The following rules apply to the use of the PHASE card:

1. A PHASE card must immediately precede each group of Call cards (CALL, CALLN, CALLP) that specify the subprograms to be included in that phase.

2. If the first PHASE card is omitted, the name of the first subprogram that is called is used by the Linkage Loader as the name of the entire multiphase program. The header record of each phase will have this name. (This is the procedure used to generate a header record for a single-phase program, which does not require a PHASE card.)

3. A PHASE card with a program name in the operand field causes the Linkage Loader to reset the relocation factor to Base Zero and to erase from the symbol table any Linkage Symbols left from the processing of previous subprograms. (This erasure of symbols can be controlled by use of the PRTCT card, which is described later.)

4. A PHASE card with a blank operand field does not reset the relocation factor, nor does it cause the erasure of any symbols from the symbol table.

5. Columns 60-70 of the PHASE card must be left blank. (These columns are used by the Linkage Loader.)

EXAMPLE.

Since the PHASE card is most commonly used in conjunction with the BASE1 card, examples illustrating the use of the PHASE card are presented immediately fol-

lowing the description of the BASE1 card.

The BASE1 Card

The BASE1 card is used to control the relocation factor. This control of the relocation factor enables the programmer to direct the Linkage Loader to relocate a phase in such a manner that it will overlay the preceding phase at a predetermined point.

Three types of operands can be used in a BASE1 card:

1. Linkage Symbol: If a Linkage Symbol appears in the operand of a BASE1 card, the Linkage Loader sets the relocation factor to the address equated with that symbol in the Linkage Loader's symbol table. For this reason, the Linkage Symbol must be defined by the processing of a subprogram previous to the one which includes the BASE1 card. In almost all cases, the Linkage Symbol is one that is declared by the Title card of a previous subprogram. This enables one phase of a program to be relocated with the same origin point used for a subprogram in a previous phase even though, at the time the BASE1 card is placed in the Standard Input Unit, the actual origin point of the previous subprogram was not yet determined. When the phase that included the BASE1 card is loaded into core storage, it will overlay the previous phase beginning at the origin point of the subprogram named in the operand of the BASE1 card.

NOTE: Since the Autocoder language offers facilities for defining Linkage Symbols other than the origin point of a subprogram, it is possible for users of Autocoder to construct phases that overlay preceding phases at locations other than the beginning of a subprogram.

2. Actual Address: If the operand of a BASE1 card contains an actual address, the Linkage Loader sets the relocation factor to that address. Particular care must be exercised in the use of this type of operand, since it is not always possible to predict the final location of program elements that preceded a BASE1 card containing an actual address.

*3. *+X00:* If the operand of a BASE1 card contains *+X00 (asterisk-plus sign-X-zero-zero), the Linkage Loader sets the current relocation factor to the next highest address that is a multiple of 100. (If the relocation factor is already at a multiple of 100, no adjustment is performed.) This type of operand is not necessarily related to a multiphase program. It is provided to control the relocation of subprograms that depend on certain areas being located at an even-hundred address. (For example, use of the Clear Storage instruction can result in this requirement.)

Each time the Linkage Loader encounters a BASE1 card, it performs the following functions:

1. It sets the relocation factor to the address value specified by the operand of the BASE1 card.

2. It erases from the symbol table all symbols that are equated to addresses higher than the address value specified by the operand of the BASE1 card. (This erasure occurs only for BASE1 cards with a symbol or actual address in the operand. The *+X00 operand does not cause symbols to be erased.) The erasure of symbols can be controlled by the PRTCT card, which is described below.

The following rules apply to the use of the BASE1 card:

1. A BASE1 card, when placed in the Standard Input Unit (rather than imbedded within a subprogram), should immediately follow a PHASE card. A BASE1 card at any point other than immediately behind a PHASE card will affect all subprograms that have not been processed, even if the calls for those subprograms preceded the BASE1 card.

2. A BASE1 card, when imbedded within a subprogram, must immediately follow the TITLE card. However, BASE1 cards with the *+X00 operand can be placed anywhere within a subprogram.

3. The address value of a Linkage Symbol used as the operand of a BASE1 card must have been defined during the processing of a previous subprogram.

EXAMPLE 1

This example illustrates the construction of a two-phase program from two subprograms. SP1 constitutes the first phase of the multiphase program, and SP2 constitutes the second phase, which is to completely overlay the first. Both subprograms are on the Library. Neither of the subprograms contains imbedded calls.

The sequence of cards in the SIU is as follows:

```
PHASE    SAM
CALL     SP1
PHASE
BASE1    SP1
CALL     SP2
```

The Linkage Loader proceeds as follows:

1. PHASE SAM: A header record is created. It contains the name SAM and the phase number 001.

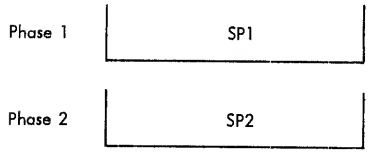
2. CALL SP1: The name SP1 is placed on the request list.

3. PHASE (blank operand): All previous calls are processed. (In this example, SP1 is the only previous call.) A termination record is generated for the first phase; a header record is generated for the second phase. It contains the name SAM and the phase number 002.

4. BASE1 SP1: The relocation factor is set to the address value established by the Title card of SP1. (All Linkage Symbols with a higher address-value are erased from the symbol table.)

5. CALL SP2: The name SP2 is placed on the request list. The Linkage Loader resumes reading from the SIU and finds no further control cards. SP2 is located and processed.

The memory map is as follows:



EXAMPLE 2

This example illustrates the construction of a two-phase program from five subprograms. SP1 and SP2 constitute the first phase, and SP3, SP4, and SP5 constitute the second phase, which is to overlay only SP2 of the first phase (leaving SP1 in storage). All the subprograms are on the Library. None of them contain imbedded calls.

The sequence of cards in the SIU is as follows:

PHASE	MATILDA
CALL	SP1
CALLN	SP2
PHASE	
BASE1	SP2
CALL	SP3
CALL	SP4
CALL	SP5

The Linkage Loader proceeds as follows:

1. PHASE MATILDA: A header record is created. It contains the name MATILDA and the phase number 001.

2. CALL SP1: The name SP1 is placed on the request list.

3. CALL SP2: In accordance with the procedure for CALLN cards, SP1 is located and processed, then SP2.

4. PHASE (blank operand): A termination record is generated for the first phase. It contains the entry point specified by the Termination card of whichever subprogram was the primary one. A header record is generated for the second phase. It contains the name MATILDA and the phase number 002.

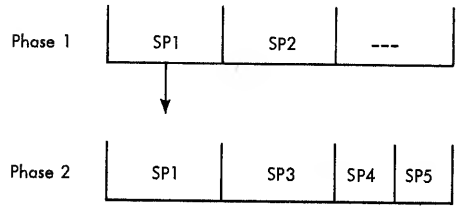
5. BASE1 SP2: The relocation factor is set to the address value established by the Title card of SP2. (All Linkage Symbols with a higher address-value are erased from the symbol table.)

6. CALL SP3: The name SP3 is placed on the request list.

7. CALL SP4: The name SP4 is placed on the request list.

8. CALL SP5: The name SP5 is placed on the request list. The Linkage Loader resumes reading from the SIU and finds no further control cards. SP3, SP4, and SP5 are located on the Library and processed.

The memory map is as follows:



EXAMPLE 3

This example illustrates the construction of two separate multiphase programs during a single run of the Linkage Loader. All subprograms are on the Library, and none of them contain imbedded calls.

The sequence of cards in the SIU is as follows:

PHASE	OLSEN	PHASE	JOHNSON
CALL	SP1	CALL	SP3
PHASE		PHASE	
BASE1	SP1	BASE1	SP3
CALL	SP2	CALL	SP4

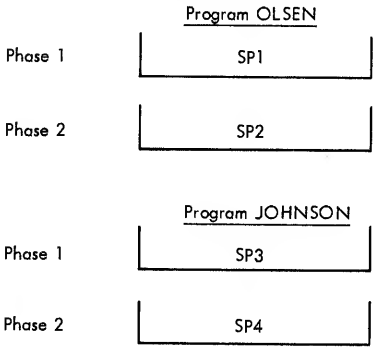
The Linkage Loader proceeds as follows:

1. PHASE OLSEN through CALL SP2: The Linkage Loader begins construction of a two-phase program in the manner illustrated by the preceding examples.

2. PHASE JOHNSON: The request list is checked for outstanding calls. The Linkage Loader finds the name SP2 on that list (from the CALL card for SP2), locates and processes SP2, and generates a termination record for the preceding phase. The relocation factor is then reset to Base Zero, and Linkage Symbols are erased from the symbol table. A header record is generated for the next phase. It contains the name JOHNSON and the phase number 001.

3. CALL SP3 through CALL SP4: The Linkage Loader constructs the second multiphase program.

The memory maps are as follows:



NOTE: The two programs are not necessarily related in any way. This example is designed to illustrate a means of directing the Linkage Loader to produce a Job File that is, in effect, a library of programs in absolute format.

The PRTCT Card

The PRTCT card is used to set a limit for erasure of Linkage Symbols from the symbol table. The Linkage Loader will retain in its symbol table all Linkage Symbols equal to or higher than the address value specified by the operand of the PRTCT card. (The operand of a PRTCT card can be either a Linkage Symbol or an actual address.) This protection will be retained until it is changed by either another PRTCT card or by the initialization that is performed each time the Linkage Loader is brought into storage for execution.

The following rules apply to the use of the PRTCT card:

1. The PRTCT card must precede any control card that would cause erasure of symbols higher than the address value specified by the operand of the PRTCT card.

2. The address value of a Linkage Symbol used as the operand of a PRTCT card must have been defined during the processing of a previous subprogram.

EXAMPLE

This example illustrates the construction of a two-phase program in which one subprogram must be placed in upper storage and be used by other subprograms in both phases. All of the subprograms are on the Library, and none of them contain imbedded calls.

The sequence of cards in the SIU is as follows:

```
PHASE   OTHELLO
CALL    SP1
CALLN   SP2
BASE1   30000
CALL    SP3
PHASE
PRTCT   30000
BASE1   SP2
CALL    SP4
```

The Linkage Loader proceeds as follows:

1. PHASE OTHELLO through CALLN SP2: SP1 and SP2 are processed as explained in previous examples.

2. BASE1 30000: The relocation factor is set to 30000. (Note that all previous calls have been satisfied because of the CALLN card. Note also that if SP2 had contained an imbedded call for a subprogram that had not yet been processed, then the new relocation factor would be applied to that subprogram.)

3. CALL SP3: The name SP3 is placed on the request list.

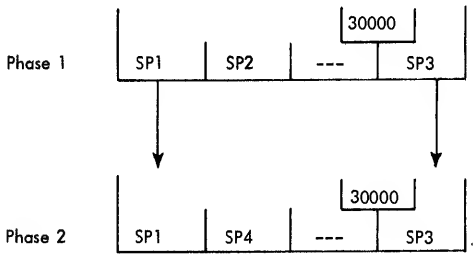
4. PHASE (blank operand): SP3 is located and processed. A termination record is generated for the first phase. A header record is generated for the second phase. It contains the name OTHELLO and the phase number 002.

5. PRTCT 30000: The Linkage Loader notes that Linkage Symbols equated to an address value of 30000 and higher are not to be erased from the symbol table. (The operand of the PRTCT card could also be SP3.)

6. BASE1 SP2: The relocation factor is reset to the address value established by the Title card of SP2. Linkage Symbols equated to an address value higher than SP2 and lower than 30000 are erased from the symbol table.

7. CALL SP4: The name SP4 is placed on the request list. The Linkage Loader resumes reading from the SIU and finds no further control cards. SP4 is located and processed.

The memory map is as follows:



The BASE2 Card

The BASE2 card is used for subprograms that refer to a common data area. The operand of the BASE2 card, which can be either a Linkage Symbol or an actual address, sets the upper limit of the common data area.

Each time the Linkage Loader encounters a BASE2 card, it sets the special relocation factor that it uses for downward relocation. This factor is then used for the adjustment of addresses that refer to the common data area.

The following rules apply to the use of the BASE2 card:

1. The BASE2 card must be positioned in the Standard Input Unit so that it is read before the processing of any subprograms affected by it.

2. More than one BASE2 card can be used, but rule 1 also applies to those cards.

3. If a Linkage Symbol is used for the operand of a BASE2 card, the address value of that symbol must

have been defined during the processing of a previous subprogram.

EXAMPLE

This example illustrates the construction of a single-phase program from three subprograms, each of which uses a common data area. All three subprograms are on the Library, and none of them contain imbedded calls.

The sequence of cards in the SIU is as follows:

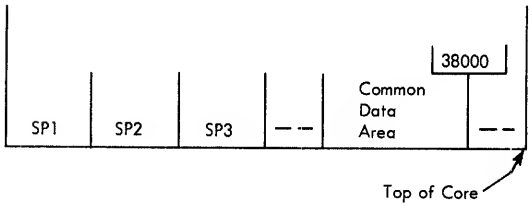
```
BASE2   38000
CALL    SP1
CALL    SP2
CALL    SP3
```

The Linkage Loader proceeds as follows:

1. BASE2 38000: The factor for downward relocation is set to 38000.

2. CALL SP1 through CALL SP3: These three subprograms are located and processed. During the processing, all references to the common data area are adjusted in accordance with the factor set by the BASE2 card.

The memory map is as follows:



The SNAP Card

The SNAP card directs the Linkage Loader to include the Snapshot Program into a specified subprogram. (The Snapshot Program is one of the Utility Programs provided in the Operating System. See the publication, *Utility Programs*, Form C28-0325, for a description of the functions of the Snapshot Program.)

The operands used for this card are as follows:

```
NAMExxxxxyyyyyzzzzzyyyyyzzzzz      56 57
                                   ss
```

Where:

NAME is the name of the subprogram,

xxxxx is the address of the instruction that is the point where the Snapshot is to be taken,

yyyyy is the address of the lower limit of the area for which the Snapshot is to be taken, and

zzzzz is the address of the upper limit of the area for which the Snapshot is to be taken.

ss Columns 56 and 57 must contain the length of the instruction at xxxxx. A leading zero must be used in the case of an instruction length of seven. The instruction at xxxxx cannot be an IOCS macro-instruction.

The addresses used as operands are the relative addresses assigned by the compiler. These addresses can be obtained from a listing of the program. The Linkage Loader will perform the normal relocation on the addresses specified in the operand field of the SNAP card.

Columns 6-10 of the SNAP card can be used to establish an identification for the Snapshot specified by the card. Any characters in those columns will be printed on the Snapshot listing.

The instruction at the point where the snapshot is to be taken must be at least seven characters in length and cannot be a chained instruction.

The following rules apply to the use of the SNAP card:

1. A SNAP card must be positioned in the Standard Input Unit so that it is read after the processing of the subprogram that it affects.

2. A SNAP card can contain either one or two sets of upper and lower limits for Snapshot areas (as indicated by the above format of the operands).

The INPUT Card

The INPUT card can be used to direct the Linkage Loader to read its control cards from a source other than the Standard Input Unit. The operand of the INPUT card is the name of the *Symbolic Unit* that contains the control cards. (Information concerning symbolic assignment of input/output units is presented under "Operation of the System Monitor.")

The following rules apply to the use of the INPUT card:

1. The name of the Symbolic Unit must be one that is currently assigned to an actual I/O unit (Physical Unit).

2. End-of-file at the specified unit causes the Linkage Loader to resume reading from the Standard Input Unit.

3. The INPUT card can contain the operand, SIU. In this case, the Linkage Loader resumes reading from the Standard Input Unit. (Such a card would, of course, be used in the alternate unit, after an INPUT card in the Standard Input Unit had directed the Linkage Loader to begin reading from that alternate unit. (*Alternate unit* should not be confused with the Alternate Input Unit.)

4. INPUT cards can be placed at any point in the control card deck for the Linkage Loader.

Programming Considerations for Use of the System Monitor

This section contains information related to the writing of programs that are to be run under control of the System Monitor. Such programs are termed *dependent programs*. Furthermore, the term *batch program* is used to refer to a program that is included in the series of jobs from the Standard (or Alternate) Input Unit (as opposed to programs included in a TELE-PROCESSING System).

Use of Linkage Sequences

Many of the facilities of the System Monitor can be utilized by dependent programs by means of program elements known as *linkage sequences*. Basically, a linkage sequence is a branch instruction followed by one or more fields of control information. The branch instruction gives control to a routine that performs a desired function in accordance with the information in the control fields. The concept and use of linkage sequences is illustrated in the following section, "Use of the Communication Region." (In this publication, the construction of linkage sequences is shown in Autocoder language statements.)

Use of the Communication Region

Contents of the Communication Region

The table below lists the areas of the Communication Region that can be addressed by dependent programs. In addition to the System Symbol that represents the low-order position of the area, the table states the length of each area and whether the contents of the area can be modified by a dependent program. Each area of the Communication Region contains a word mark only in its high-order position.

It is important to note that *addressing* an area of the Communication Region does not imply *modifying* the contents of that area. For addressing an area, the dependent program contains an instruction that refers to the area only in order to determine its current contents. For modifying an area, the dependent program contains a linkage sequence that specifies new information to be placed into the area by a routine in the Resident Monitor. (The linkage sequence for modifying an area is described below, following the table.)

SYSTEM SYMBOL	CONTENTS OF AREA	LENGTH	MODIFIABLE
/AMS/	Actual machine size (highest addressable location). This area is set at System Generation.	5	No
/CRD/	The high-order address of the input area filled by the Read Routine of the Resident Monitor. (This routine, described in a later section, is the unit-record routine that reads from the Standard Input Unit.)	5	No
/DAT/	Date (first two positions for the year, followed by three positions for the day). This area is set during the initialization of the Resident Monitor (see Appendix).	5	No
/IPI/	Inter-program information. (This area can be used to store control information between runs within a job. For example, a program can set switches in this area that can be tested by the next program executed.) This area is set to blanks between each job.	5	Yes
/LIN/	Number of lines per page for printer output. (This area is set by any program that wishes to establish a constant that can be used to compare to a program counter for writing printer output.)	2	Yes
/ORG/	Origin point for batch programs. (This area, which is set at System Generation, is used by the Linkage Loader to determine the value of Base Zero.)	5	No
/PHN/	Phase number of program phase currently being executed. (This area is set to 001 between each run.)	3	No

SYSTEM SYMBOL	CONTENTS OF AREA	LENGTH	MODIFIABLE
/PNM/	Name of program currently being executed. (The name is left-justified in the area.)	10	Yes
/SIZ/	Highest location loaded for the current batch program (including any previous phases of the current batch program). This area is set by the Load Routine of the Resident Monitor. NOTE: The TELE-PROCESSING Supervisor checks this area to determine the amount of core storage currently available for programs required to process input from a TELE-PROCESSING device. Therefore, if a batch program uses locations above those that were loaded, that program should modify this area of the Communication Region to reflect the highest location used.	5	Yes
/TPB/	Lowest location of the area reserved for programs under control of the TELE-PROCESSING Supervisor. (TELE-PROCESSING Systems are described in the publication, *The* TELE-PROCESSING *Supervisor*, Form C-28-0321.)	5	No

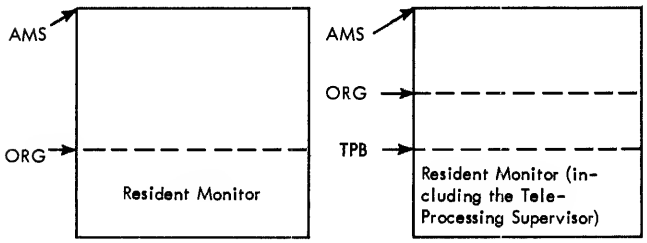


Figure 2. Relationship of the AMS, ORG, and TPB Fields

Figure 2 illustrates the relationship of the core-storage locations represented by the AMS, ORG, and TPB fields of the Communication Region. The diagram at the left side of Figure 2 illustrates these locations for core storage that does *not* include an area reserved for programs under control of the TELE-PROCESSING Supervisor. The other diagram in Figure 2 illustrates these locations for core storage that *does* include such a reserved area.

Modification of the Communication Region

To modify an area of the Communication Region, the dependent program must contain a linkage sequence constructed as follows:

```
BXPA    /MCR/
DCW     /xxx/
DCW     yyyyy
DCW     zzzzz
        (Next sequential instruction in dependent program)
```

In this linkage sequence, /xxx/ represents the System Symbol of the area that is to be modified, yyyyy is the low-order address of the new information to be placed in that area, and zzzzz is the address to which control is to be returned in the event the linkage sequence attempted to modify an area of the Communication Region that cannot be modified. The routine that modifies the Communication Region checks each modification request to determine whether the System Symbol /xxx/ refers to a modifiable area. If it does not, control is returned to the dependent program at location zzzzz; if it does refer to a modifiable area, the modification is performed and control is returned to the next sequential instruction following the linkage sequence that issued the modification request.

Use of the Unit-Record Routines

The three unit-record routines in the Resident Monitor can be used by dependent programs. One of these routines reads input from the Standard Input Unit (or Alternate Input Unit), another writes output on the Standard Print Unit, and the third writes output on the Standard Punch Unit. (Each of these units can be either a unit-record device or a tape unit, as specified at System Generation.)

The dependent program addresses these routines by means of a linkage sequence. The branch instruction of the linkage sequence uses a System Symbol to refer to the entry point of each of the routines.

All three routines perform their functions with unblocked records in the MOVE mode. If tape is used for any of the routines, the unit-record functions are performed in ODD parity.

Read Routine

The Read Routine reads 80-character records from the Standard Input Unit. (This unit could be the Alternate Input Unit if the Resident Monitor has been instructed to change from one unit to the other.)

The high-order address of the input area used by the Read Routine is located in the Communication Region. (The input area is in the Resident Monitor.) The System Symbol /CRD/ is used to refer to the area of the Communication Region used for this purpose. (Word marks must not be set in the input area.)

The Read Routine checks each input record to determine whether it is a Monitor control card. If it is, the

This section contains information concerning the machine-room operation of the System Monitor. It is primarily directed to machine-room personnel, but much of the material, such as the description of control cards required to define a program job, will also be of interest to programmers.

Initialization of the System Monitor

The first element on the System Operating File is a routine that brings the Resident Monitor and Transitional Monitor into core storage and performs the necessary initialization functions. This routine (which is called the Bootstrap) can also be used if re-initialization of the System Monitor becomes necessary because of an emergency situation. (For example, a dependent program might fail in such a way that portions of the Resident Monitor are destroyed.) Re-initialization procedures include facilities for repositioning the Standard Input Unit (if tape) to the next job specified by the operator.

During initialization (or re-initialization) procedures, the installation supplies the Resident Monitor with "daily information", which includes the current date and assignments of input/output units for certain files used by the System Monitor. (The Appendix contains a sample control-card deck, which includes cards used for daily information.)

Procedures for using the Bootstrap depend upon the type of device used for the System Operating File (tape or disk) and the type of device used for the Standard Input Unit (tape or card reader). The following information outlines the procedures for the various possible configurations.

SOF Is Tape; SIU Is Card Reader

Both initialization and re-initialization are effected by loading the Bootstrap into storage from the SOF. The Bootstrap reads the daily information from the SIU.

SOF Is Tape; SIU Is Tape

Both initialization and re-initialization are effected by loading the Bootstrap into storage from the SOF. During initialization, the Bootstrap reads the daily information from the SIU. During re-initialization, the daily information can be entered through the console, or it can be read by the Bootstrap from the SIU (which must have been rewound to load point).

SOF Is Disk; SIU Is Tape or Card Reader

For installations that use disk storage for the SOF, the Bootstrap is divided into two sections: a Preliminary Bootstrap and a Main Bootstrap. The Preliminary Bootstrap, which must be read from the SIU, provides the means for loading the Main Bootstrap from the disk SOF.

For initialization, the Preliminary Bootstrap is entered through the SIU and loads the Main Bootstrap from the SOF. The Main Bootstrap reads the daily information from the SIU. It places the daily information into the proper areas of the System Monitor and also stores that information into the disk SOF. (This is done to simplify subsequent re-initialization, if required.)

For re-initialization, the operator can manually branch to the Preliminary Bootstrap (which remains in the Resident Monitor from the initialization procedure). If this area of the Resident Monitor has been destroyed, the operator can either enter the Preliminary Bootstrap from the console, or load it from the SIU. The Preliminary Bootstrap loads the Main Bootstrap from the disk SOF, and the Main Bootstrap loads and initializes the System Monitor. Daily information is already contained in the System Monitor being loaded from the SOF, since it was placed there at the time of initialization.

Control Cards for the System Monitor

This topic describes the control cards that can be placed in the Standard (or Alternate) Input Unit to direct the operation of the System Monitor. At the time of System Generation, the installation can specify that the System Monitor is to record each of the Monitor control cards on the console typewriter and/or the Standard Print Unit.

Format

The format of control cards for the System Monitor is based on the format used for Autocoder source statements:

Columns 6-10 (first portion of the *Label field*) contain the characters MON$$ to identify the card as one directed to the System Monitor.

Columns 16-20 (the *Operation field*) are used to identify the function of the card. (The mnemonics used for this area of the card are presented in the following list of Monitor control cards.)

Columns 21-72 (the *Operand field*) are used to specify additional information related to the function specified in columns 16-20. The standard Autocoder rules for operands apply to information in this portion of a Monitor control card: operands must be separated by a comma; operands cannot contain blanks; an intentionally omitted operand must be indicated by placing its trailing comma adjacent to the preceding comma (except in cases where the omitted operand is the last operand).

Names and Functions of Monitor Control Cards

The following list contains the control cards that are directed to the System Monitor. Control cards that supply information related to the processing of a particular program within the Operating System, such as the Storage Print Program, are presented in the publication that describes that program. Control cards for the Linkage Loader are described in an earlier section of this publication.

In the descriptions below, each type of Monitor control card is identified by the mnemonic immediately following the numeral. The mnemonic is to be punched in columns 16-20. If it contains less than five characters, it must begin in column 16; unused columns in this field are left blank. Following the mnemonic for each card are the operands that are used for that card.

JOB NAME

This card is used to indicate the beginning of each job. The operand is the name assigned to that job by the user. Each time a JOB card is encountered, the Resident Monitor spaces the Standard Print Unit to the next page and prints the contents of the JOB card as the first line on that page. (If the Standard Print Unit is tape, the proper carriage control character is placed into the output record.) The Resident Monitor prints either an "S" or an "A" on the line with the JOB card to indicate whether it was read from the Standard or the Alternate Input Unit. In addition, the Resident Monitor places a two-character sequence number on the line with the JOB card. This sequence number indicates the number of JOB cards that have been read since the beginning of the batch. (See the description of the END card for details concerning the JOB card sequence number.)

The contents of the JOB card (and the additional information supplied by the Resident Monitor) are always written on the console typewriter and the Standard Print Unit, regardless of whether other Monitor control cards are also printed and/or typed. An additional feature that can be specified at the time of System Generation is the recording of JOB cards on the Standard Punch Unit.

| MODE GO,TEST

The MODE card is used to supply information related to the type of job to be performed. If the operand GO is specified, the output from following compilations will be written on the Go File for subsequent execution. The operand TEST is used to specify that the following dependent program is being tested. These operands set indicators within the Resident Monitor. The indicators are reset by the next JOB card.

EXEQ PROGRAMID,MJB,INPUTDATA

The EXEQ card is used to request the execution of the program named in the first operand. (The name of the program is placed into the PNM field of the Communication Region.) The second operand specifies the file containing that program. This operand can be either SOF for the System Operating File, or MJB for the Job File. (The absence of a second operand indicates that the program is on the SOF; therefore, it is not necessary to punch the characters SOF. The omission of SOF must be indicated by a comma if succeeding operands are used.)

The third operand, which is optional, specifies the location of the input data for the requested program. This operand causes the Read Routine of the Resident Monitor to read from the unit represented by the operand. In effect, the Read Routine treats the specified unit as an Alternate Input Unit. Therefore, this operand can be used only if the facility for reading an Alternate Input Unit was incorporated into the Resident Monitor at the time of System Generation. Furthermore, this operand cannot be used in an EXEQ card that is placed in the Alternate Input

an alternate cannot be shared by another Symbolic Unit as a base unit. (For example, these two assignments cannot exist simultaneously: MR1, A1, A2 and MR2, A2.)

However, some jobs may require that a Physical Unit, assigned to a Symbolic Unit as an alternate, be assigned as the base unit of another Symbolic Unit later in the job (or vice versa). For instance, assume that the first run in a job is the execution of a program that requires that Symbolic Unit MR1 be assigned to the Physical Units A1, A2, and A3 (MR1, A1, A2, A3). The next run in the job is a compilation, and A2 must be used as a Work Unit for the compiler. (The program executed in the first run of the job did not leave any valuable information on A2.) Since A2 was assigned as an *alternate* Reserve Unit for the first run, it cannot be assigned as a *base* Work Unit for the compiler, unless the initial assignment is cancelled. (Although Reserve Units can be reassigned between runs within a job, their assignments are cancelled by the System Monitor only at the end of the entire job.) The following information describes the method for cancelling assignments in order to meet the requirements of jobs such as the one just described.

CANCELLATION OF ASSIGNMENT

Assignments for Symbolic Units can be cancelled by means of an ASGN card containing only the first operand. For example, the ASGN card in Figure 5 cancels the assignment for the Reserve Unit discussed in the example above. This permits the Physical Unit that was previously assigned to that Reserve Unit to be used for the Work Unit required for the compiler.

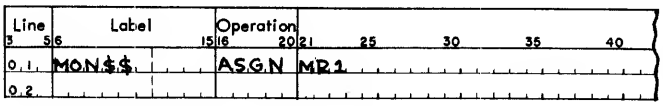


Figure 5. Cancellation of Assignment

The ASGN card in Figure 5 and the ASGN card used to assign A2 to the compiler's Work Unit should both be placed immediately before the EXEQ card that requests execution of the compiler. (The card used to cancel the assignment to MR1 must precede the card used to make the assignment to the Work Unit.)

PLACEMENT OF THE ASGN CARD

The ASGN cards, like all other Monitor control cards, are submitted to the System Monitor through the Standard (or Alternate) Input Unit. The relative position of an ASGN card within the control-card deck is determined by the category of Symbolic Unit for which the ASGN card is being used. As described earlier in this topic, Symbolic Units are divided into five categories with respect to the time they can be assigned. (Refer to the information under the heading, "Assignment Times for Symbolic Units.") ASGN cards for Symbolic Units that are assigned during initialization or re-initialization procedures are included as part of the daily information in the Standard Input Unit. ASGN cards for Symbolic Units that are assigned only for the duration of a job must follow the related Job Card and precede the EXEQ card for the program that requires the assignments. ASGN cards for Symbolic Units that are used by the TELE-PROCESSING System must precede the control card that opens the TELE-PROCESSING System.

Use of the ASGN Card for System Units

The Physical Units assigned for the use of certain System Units cannot be assigned to any other Symbolic Units during the time they are assigned to those System Units. Those System Units are as follow:

System Operating File (MSOF/)
Standard Input Unit (MSIU/)
Standard Print Unit (MSPR/)
Standard Punch Unit (MSPU/)
System Library File (/LIB/)
Core Image File (/MDM/)

In addition, Physical Units that are assigned to Symbolic Units used by the TELE-PROCESSING System cannot be assigned to any other Symbolic Units during the time the TELE-PROCESSING System is ready to receive input from TELE-PROCESSING devices.

It is important to note that the Physical Units assigned to the System Operating File and the Standard Input Unit can be released for use by other Symbolic Units only through the procedure of System Generation. The Physical Units assigned to the Standard Print Unit, the Standard Punch Unit, and the Core Image File can be released for use by other Symbolic Units only through the procedure of initialization or re-initialization of the System Monitor. The Physical Unit assigned to the System Library File (if this file is not a part of the System Operating File) can be released for use by other Symbolic Units either through initialization and re-initialization procedures, or through the cancellation of the Library's assignment by means of an ASGN card containing LIB as the only operand.

Assignment of the Go File

For compile-and-go operations using AUTOCODER, COBOL or FORTRAN, the Go File must be assigned to a Physical Unit other than those used for the compiler's work files.

Console Inquiries

Console Inquiries are used to communicate control information to various portions of the System Monitor. The messages that can be entered through the console are divided into three groups, as follows:

A. Messages that can be accepted by the Resident Monitor during the execution of a dependent program.

B. Messages than can be accepted by the Transitional Monitor when it is in storage to perform transition from one program to the next.

C. Messages that are communicated to the Transitional Monitor in reply to a request from the Transitional Monitor for specific control information.

The various messages are communicated to the System Monitor in the following manner:

1. Depress the Console Inquiry key.
2. Type in the message .
3. Depress the Inquiry Releasc key.

Group A Messages

The following messages can be accepted during the execution of a dependent program. Unless otherwise stated, the entire message consists of the three characters that are listed.

$10 This message causes immediate entry to the Unusual End-of-Program portion of the Resident Monitor's End-of-Program Routine. (See previous sections of this publication for information concerning this routine.) This message can be used only for batch programs (not programs operating under control of the TELE-PROCESSING Supervisor), and can be entered at any time.

NOTE: The operator can also give control to the Unusual End-of-Program procedures by pressing the Computer Reset key and then the Start key. It is important to note, however, that this procedure resets certain machine indicators. Therefore, the records written on the Core Image File will not reflect the status of core storage at the exact time the dependent program failed.

$20 This message causes the Resident Monitor to set a program switch for the Transitional Monitor, indicating that the Transitional Monitor is to notify the operator when it is ready to accept Console Inquiries. (Messages accepted by the Transitional Monitor are described below, under the heading "Group B Messages.") This message can be entered at any time.

$3n This message allows the user to exercise a program option as the program is being executed, but it should be limited to situations that cannot be handled by control cards. Thus, the $3n message closely approximates external toggle switches.

Upon receiving this message, the Resident Monitor moves the single character "n" to the one-character field /MCI/. The user's program must interrogate the character at /MCI/ and take the desired action. The character may be any valid BCD character.

The message may be used to enter more information than the single character. However, after recognizing a particular character at /MCI/, the user's program must determine that the input area still contains the message by checking for "$3n" in the input area. (The high-order position of the input area is designated /RIQ/.) The check is necessary because no protection is given to the data in the input area. It is advisable to move the data before another console message overlays it.

The Resident Monitor restores /MCI/ to a blank only at End of Program and at Unusual End of Program. The user may request the setting of /MCI/ to a blank by the following sequence:

```
BXPA    /MCR/
DCW     /MCI/
DCW     XXXXX (address of a blank)
DCW     ERR
```

This sequence is covered in a previous topic, "Modification of the Communication Region."

For example, the sequence:

```
BXPA    /MCR/
DCW     /MCI/
DCW     XXXXX (address of a blank)
DCW     ERR
IOCTL   TYPE, MESSAGE
BCE     *-11, /MCI/,b
```

could be used when the "MESSAGE" will direct that an external decision be made. The branch character equal instruction will loop to itself until the character has been entered by the $3n message.

$50 This message causes the Resident Monitor to exit from its Wait-Loop Routine and return control to the dependent program. (Information concerning the Wait-Loop Routine is presented in the section "Programming Considerations for Use of the System Monitor.") This message can be entered at any time the dependent program indicates that it is using the Wait-Loop Routine.

$70 This mcssage signals the Resident Monitor to initiate Immediate Restart procedures. (For information concerning checkpoint-and-restart procedures, see a later topic of this section, "Restarting from a Checkpoint.")

$90b This message is used to communicate information to the TELE-PROCESSING System. Messages for the TELE-PROCESSING System can be up to twenty characters in total length and can be entered at any time. (See the publication, *Tele-Processing Supervisor,* for detailed information concerning these Console Inquiries.)

Group B Messages

The following messages can be accepted during the time the Transitional Monitor is in core storage. The Transitional Monitor notifies the operator when group B messages can be accepted by means of a console message. This message is written if the operator previously requested it by entering the $20 message (described above). The Transitional Monitor also notifies the operator it is ready to accept group B messages each time an END card is read from the Standard Input Unit. After the Transitional Monitor notifies the operator, it enters a wait-loop. Unless otherwise stated, each of the following messages consists entirely of the three characters that are listed.

$B1 This message causes the Read Routine of the Resident Monitor to be altered to read from the Alternate Input Unit. This message can only be given if the $20 message has been previously entered. It cannot be given if the Transitional Monitor is in a wait-loop caused by an END card from the Standard Input Unit. (See the $BX message, which is described below, for procedures concerning the END card from the Standard Input Unit.)

$B2 This message causes the Transitional Monitor to close, rewind, and unload the tape file designated as the Standard Print Unit. The operator then mounts another tape on that unit.

$B3 This message causes the Transitional Monitor to close, rewind, and unload the tape file designated as the Standard Punch Unit. The operator then mounts another tape on that unit. (This message is used only if the Standard Punch Unit is not assigned to the same tape as the Standard Print Unit. If the two units are assigned to the same tape, then the $B2 message is used to perform the closing functions.)

$B4 This message causes the Transitional Monitor to close, rewind, and unload the tape file designated as the Core Image File (the file that contains records of core storage used for restarting from checkpoints or for obtaining Storage Prints). The operator then mounts another tape on the unit used for this file. (This message is applicable only if the Core Image File, which is optional, had been specified for the installation at the time of System Generation.)

$B5xx This five-character message indicates to the Transitional Monitor that the Assignment Symbol specified by xx is currently unavailable for use. (The two-character symbol represented by xx in this description is one of the symbols assigned by the installation to input/output units.)

$B6xx This five-character message indicates to the Transitional Monitor that the Assignment Symbol specified by xx is now available for use. The Transitional Monitor remove the *unavailable* indication that it had set in response to a $B5xx message for this unit (see above).

$BX This message indicates that the operator has no further group B messages at this time, and causes the Transitional Monitor to exit from the wait-loop that it had entered to enable the operator to make Console Inquiries. (After performing the functions indicated by each of the other group B messages, the Transitional Monitor returns to the wait-loop for further messages. Therefore, the $BX must be given to enable the Transitional Monitor to resume its processing.)

After the $BX message is entered, the Transitional Monitor completes functions related to previous group B messages. For example, if the $B2 message had been given to close the Standard Print Unit, the Transitional Monitor opens the new one after the $BX message is given. The complete procedure for such a function is as follows:

1. The Transitional Monitor notifies the operator it is ready to accept group B messages and then enters a wait-loop.
2. The operator enters the $B2 message.
3. The Transitional Monitor performs the closing functions for the Standard Print Unit and returns to the wait-loop.
4. The operator mounts another tape for the Standard Print Unit and enters the $BX message.
5. The Transitional Monitor opens the file for the new tape and resumes other processing related to preparation for the next program (such as processing the JOB card for that program).

As indicated in the description of the $B1 message (described above), the Transitional Monitor enters a wait-loop each time it encounters an END card in the Standard Input Unit. When this wait-loop is terminated by the $BX message, the Read Routine of the Resident Monitor is still set to read from the Standard Input Unit. In order to change this routine to read from the Alternate Input Unit, the operator must have previously entered the $20 message.

Group C Messages

The following message is given in reply to a message from the Transitional Monitor. The message from the Transitional Monitor notifies the operator that an ASGN card has been read for an Assignment Symbol that is currently unavailable (as indicated by a previous $B5xx message).

$C1ss This message specifies that the Assignment Symbol represented by ss is to be substituted for the one contained in the ASGN card. The substitute unit must be one that is currently available. (The

$B6xx message cannot be used at this time to indicate that the unit specified on the ASGN card is now available; it must have been given previous to the time the ASGN card is read.)

Restarting from a Checkpoint

The System Monitor provides facilities to restart programs that have been interrupted because of certain error conditions recognized by the operator or because a program of higher priority required the machine. (Also, the processing of some programs requires such a great amount of time that the processing is performed in installments. Such programs can utilize the restart facilities of the System Monitor if checkpoints are taken by that program.)

Immediate and Delayed Restarts

The Resident IOCS provides a routine for writing checkpoint records on the Core Image File. (See the publication, *Basic IOCS*, Form C28-0318, for detailed information concerning the creation of checkpoint records.) These records consist of the information in core storage at the time the checkpoint is taken. From these records the System Monitor can restart a program in one of two ways:

Immediate Restart: An Immediate Restart can be effected by entering the $70 message from the console. This message causes the Resident Monitor to bring the Transitional Monitor into storage. The Transitional Monitor then locates the most recent checkpoint on the Core Image File and initiates a restart of the dependent program from that checkpoint. The major functions performed for the Immediate Restart are as follows:

1. Reloading the dependent program into core storage.
2. Restoring areas of the Resident Monitor related to the status of the dependent program at the time the checkpoint was taken.
3. Repositioning some tape files that were open at the time the checkpoint was taken.
4. Resuming execution of the dependent program at the point at which the checkpoint was taken.

Delayed Restart: A Delayed Restart is effected by initialization of the System Monitor. At the time of an initialization for the purpose of a Delayed Restart, the operator supplies control information specifying the particular checkpoint from which the dependent program is to be restarted. (Note the difference from the Immediate Restart, which uses only the most recent checkpoint.)

The functions performed for the dependent program during a Delayed Restart are essentially the same as those performed during an Immediate Restart. Care must be exercised, however, that the configuration of input/output units and files that existed at the time the checkpoint was taken is duplicated for the restart procedures.

Restart Considerations

The following considerations apply to dependent programs using the checkpoint-and-restart facilities of the System Monitor:

1. No re-positioning can be performed for unit-record equipment. (Therefore, the Standard Input Unit must be tape.)
2. Programs that perform updating functions on records in disk storage cannot be effectively restarted, since there is no means available to the restart procedure to restore disk records that have been changed.
3. Tape files that are closed at the time of a checkpoint cannot be repositioned during a restart from that checkpoint. (This presents no problem if the tape file had been rewound at the time it was closed.)
4. Programs operating under control of the TELEPROCESSING Supervisor cannot use the checkpoint-and-restart facilities.

Units	Initial-ization	First Job	Second Job	Third Job	Fourth Job
A1	SOF	SOF	SOF	SOF	SOF
A2	MW1	MW1 and MR1	MW1 and MJB	1. MW1 cancelled 2. MR4 (base unit) 3. MR4 cancelled 4. MRB	MW1
A3	MW2	MW2 and MR3	MW2 and MR8	1. MW2 cancelled 2. MR4 (alternate unit) 3. MR4 cancelled 4. MRC	MW2
B1	MDM	MDM	MDM	MDM	MDM
B2	MW3	MW3 and MR2	1. MW3 2. MW3 cancelled 3. MR9 (base unit)	MR5 (base unit)	MW3
B3	MW4	MW4 and MGO	1. MW4 and MGO 2. MW4 and MGO cancelled 3. MR9 (alternate unit)	MR5 (alternate unit)	MW4
B4	un-assigned	MJB	source program tape for Autocoder	MJB and MRA	unas-signed
C1	SIU	SIU	SIU	SIU	SIU
C2	SPU	SPU	SPU	SPU	SPU
C3	SPR	SPR	SPR	SPR	SPR

Figure 8. Sample Assignments of Input/Output Units

Index